Controlling Complex Systems Dynamics without Prior Model

Jérémy Boes, François Gatto, Pierre Glize, Frédéric Migeon
Institut de Recherche en Informatique de Toulouse, Université Paul Sabatier, Toulouse, France
{*boes, gatto, glize, migeon*}*@irit.fr*

Keywords: Complex Systems control ; Multi-Agent Systems ; Self-Organization

Abstract: Controlling complex systems imposes to deal with high dynamics, non-linearity and multiple interdependencies. To handle these difficulties we can either build analytic models of the process to control, or enable the controller to learn how the process behaves. Adaptive Multi-Agent Systems (AMAS) are able to learn and adapt themselves to their environment thanks to the cooperative self-organization of their agents. A change in the organization of the agents results in a change of the emergent function. Thus we assume that AMAS are a good alternative for complex systems control, reuniting learning, adaptivity, robustness and genericity. The problem of control leads to a specific architecture presented in this paper.

1 INTRODUCTION

It is well known that the more complex a system is, the more complex its controler has to be (Ashby, 1956). Over the years, the number systems we intend to control, as long as their complexity, has been growing up. The need for controllers able to deal with this complexity is becoming prominent.

Whether it is car engines, bioprocesses or energy management systems or any other example, their associated controller have to deal with the same issues. Firstly, the controller needs to deal with the complexity of the controlled system. This requires the method it uses to learn and adjust its control in order to scale to the controlled system complexity. Secondly, observable and controllable points of complex systems can vary over time. Some points may be added when the system is expending as other be deleted due to internal sensor failures for example. Therefore the controller should be able to modify, preferably at runtime, its inputs and outputs. Finally, one of the primary drawbacks of current controllers is the long instantiation work they require to make them fit the actual process they will control. This is mainly due to the models they use. Genericity and adaptivity are needed in order to avoid the use of a model. The controller we present in this paper intends to tackle those issues.

Humans fulfill the task of control in their everyday life. Children learning to drive a bike is a good example. They have the ability to move their legs in order to walk and run. Also they do not have any knowledge about the mechanism inside the bike nor about the laws of physics. While they are experiencing their first ride on a bike, their most used skill is the ability to adapt and coordinate their movement in a new situation. They try new movements and will learn from every one of them. Following a practice period, children usually end up acquiring the ability to control the bike, but still did not get in touch with any theoretical concept and did not build any analytical model of the bike. This analogy tries to explain the point of view we adopted.

In section 2 various control methods and approaches that have been experimented and used over the years are briefly presented. We follow with an introduction to the Adaptive Multi-Agent Systems theory. In section 3 the design of our controller is described before going deeper in the agents behavior before section 4 concludes with our perspectives.

2 RELATED WORKS

In this section the main approaches of control are presented before a brief introduction to the Adaptive Multi-Agent Systems theory.

2.1 Complex Systems Control

Controlling systems is a generic problem that can be expressed as finding which modifications are needed to be applied on the inputs in order to obtain the desired effects on the ouputs. The most well-known are presented in the next paragraphs.

PID - The widely used Proportional-Integral-Derivative (PID) controller computes three terms related to the error between the current and the desired state of the process, from which it deduces the next action to apply (Astrom and Hagglund, 1995). PID controllers are not efficient with non-linear systems and can only handle one input, which is a severe drawback for complex systems control.

Adaptive Control - Model-based approaches like Model Predictive Control (MPC) (Nikolaou, 2001) use a model able to forecast the behavior of the process in order to find the optimal control scheme. These approaches handle several inputs but are limited by the mathematical models they use. The Dual Control Theory uses two types of commands : the actual controls that drive the process to the desired state, and probes to observe the process reactions and refine the controller's knowledge (Feldbaum, 1961). The concept of this approach is interestin but a heavy instantiation work is still required.

Intelligent Control - Intelligent control regroups approaches that use Artificial Intelligence methods to enhance existing controllers. Among these methods we can find neural networks (Hagan et al., 2002), fuzzy logic (Lee, 1990), expert systems (Stengel, 1991) and bayesian controllers (Colosimo and Del Castillo, 2007). These methods can be easily combined one with another.

2.2 Adaptive Multi-Agent Systems

The Adaptive Multi-Agent Systems (AMAS) theory is a basis for the design of multi-agent systems where cooperation is the engine for self-organization (Georgé et al., 2011). As cooperative entities, AMAS agents try to reach their own goals as well as they try to help other agents to achieve their goals. Moreover, an agent will modify its behavior if it thinks that its actions are useless or detrimental to its environment. Such situations are called Non-Cooperative Situations (NCS). Some behavioral rules, specific to NCS's, help agents to solve or avoid these situations. By solving NCS's, in regard to their own local goals, cooperative agents collectively find a solution to the global problem. Therefore one can consider the behavior of an AMAS as emergent.

Thanks to its adaptiveness, an AMAS-based controller should not rely on a specific model of the process thus it does not need a heavy instantiation work. Besides, it should be able to deal with a changing number of inputs and outputs.

3 CONTROLLER OVERVIEW

Controlling a system means finding the most adequate action to apply on its inputs in order to obtain the desired effect on its outputs. Here we present the required basic abilities of a complex system controller, and what are the agents that enable them. Then, we explain with a simple example how it is possible to control a process with local behavioral rules.

3.1 Nominal Behavior

The next paragraphs describe how our multi-agent system, called Self-Organizing Controller (SOC), works when it is already adapted to the controlled system. The mechanisms that lead to this adaptation will be explained further.

3.1.1 Observing the Process

If we intend to control a system, it is obvious that we need to be able to observe it. A specific agent type is in charge of perception, called *Variable Agent* (each input and output of the process). These agents perceive their value from the process and send it to agents who need this information. Also, Variable Agents can embed noise reduction algorithms if this problem is not handled by a third party system.

3.1.2 Representing Objectives and Constraints

The controller needs to know what is the desired state of the process. This state is represented by a set of *Constraint Agents* and possibly by additional Variable Agents.

There are three types of Constraint Agents : Threshold, Setpoint and Optimization. A Threshold Constraint Agent expresses the will to keep a variable either below or above a threshold specified by a Variable Agent. In a similar way, a Setpoint Constraint Agent expresses the will to set a process variable to a particular value. Finally, an Optimization Constraint Agent represents the will to minimize or maximize a process variable.

Each Constraint Agent computes a *critical level* that varies from 0 (the agent is satisfied) to 100 (the agent is far from satisfied). The critical level depends on the value of the variable on which the constraint is applied and can be parametrized by a second Variable (in the case of a threshold or a setpoint constraint).

It is clear that decreasing the critical levels means solving the constraints, and the only way to do so is to perform the adequate actions on the process inputs. Finding these actions implies to be able to analyze the current sate of the environment.

3.1.3 Analyzing the State of the Environment

The SOC environment is the process to control as well as the user-defined setpoints and thresholds. Thanks to Variable Agents and Constraints Agents, the multi-agent system has a representation of its environment. Before it can perform control, it must be able to extract relevant information from this representation. This is the role of agents called *Context Agents*.

A Context Agent memorizes the effects, on every critical level, of an action applied to one particular input of the process. It also memorizes the state of the environment when the action was applied. To represent this state the Context Agent maintains a set of *validity ranges*, containing one range per Variable Agent. The memorized effects on critical levels form its *forecasts*. In other words, a Context Agent represents the information that *if every variable value is inside its validity range, and if this action is applied, then the effects on every critical level will be similar to these forecasts*.

A Context Agent is said *valid* when the environment is in a state that matches its validity ranges. When this occurs, it sends a notification with its action and forecasts to the appropriate *Controller Agent*, which will be presented in the next part.

3.1.4 Selecting the Adequate Action

Each controlled input of the process is associated with a Controller Agent. The role of a Controller Agent is to apply the most adequate action in order to reduce the critical levels. It will base its choice on the information it receives from Context Agents, picking the action that will provoke the biggest descrease of the critical levels. When an action is picked, the Controller Agent notifies every Context Agents who proposed it.

Figure 1 shows the global architecture of the system. There are several cases where the Controller Agent is unable to make a good decision, because of incomplete or incorrect information from Context Agents. These cases are *Non-Cooperative Situations (NCS)*. When a NCS occurs, the cooperative behavior of involved agents is triggered in order to solve it. This will be explained in 3.2.

3.2 Non-Cooperative Situations

This section presents the main NCS's our agents face and how they solve it, leading the system to have an accurate representation of the process to control.

No Adequate Action in Suggestions - This NCS occurs when the suggestions list of a Controller Agent contains only forecasts of increasing Constraint Agents critical levels. There are two cases : either all the possible actions are already suggested or some actions are not proposed. In the first case, the only choice left is to accept the suggestion with the less bad forecasts. In the second case, a new action is applied, and a new Context is created with this action.

Empty Suggestions List - This NCS happens when a Controller Agent has to apply an action, but finds its suggestion list empty. It will be unable to find an adequate action with certainty, but it can make some hypothesis to try one. If the last action it applied had reduced the maximum of the Constraint Agents critical levels, the same action is reproduced. If not, the opposite action is applied. A new Context Agent is created, with the applied action. After its creation, a Context Agent will extend its validity ranges as long as its action is applied.

Wrong Forecast - This NCS occurs when a Context Agent is selected, checks its forecasts and notices that they are not correct. If the forecasts are wrong (i.e. a critical level evolves in the opposite direction of the forecast), the Context Agent considers that it should not have been valid when the action was suggested : its validity ranges are reduced. A Context Agent dies if one of its ranges is reduced to an amplitude of zero. If the forecasts are only erroneous (i.e all the critical levels evolve in the forecasted direction, but not with the forecasted amplitude), the Context Agent considers that its validity was relevant, thus does not modify its validity ranges, but adjusts its forecasts to match its observation.

4 CONCLUSION

In this paper, we presented a an Adaptive Multi-Agent System, called SOC, which controls an undefined complex system, basing its behavior on self-organizing agents, without a complete understanding of the controlled process. All the controller needs to know is how the process behaves. In other words our controller is of black-box type : the multi-agent system only perceives the process inputs and outputs, but not its internal mechanisms. The basic principle of our controller is the following : the multi-agent system memorizes the state of the process inputs/outputs when an action is applied and observes the reactions of the process. It will use this information to decide whether this action was good or not in regard of the user-defined desired process state. This means that the quality of the control improves over time : at the begining the controller knows nothing about the process, but it perpetually learns from its actions and

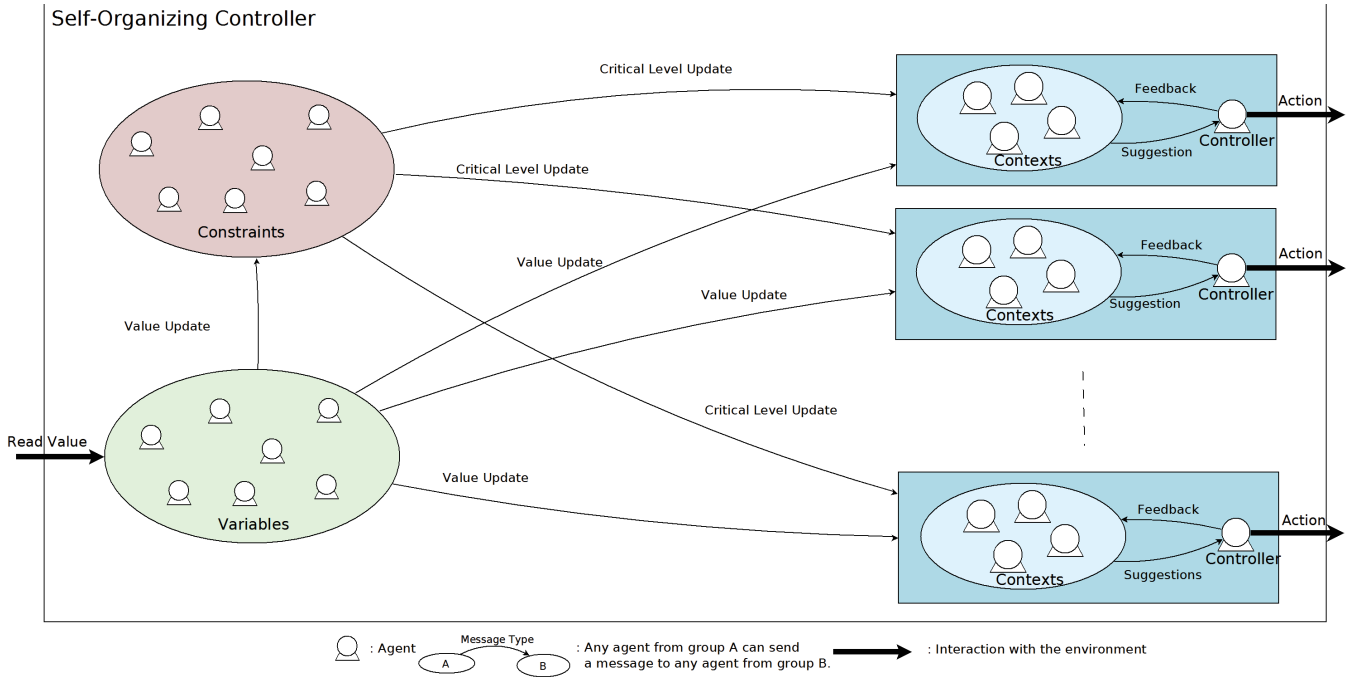

Figure 1: SOC Topology

quickly manages to control the process. Since the learning is parallel to the control, SOC continuously self-adapts to the process.

SOC does not need any prerequisite knowledge other than the intentions of the user (i.e. some criticality functions). It is also able to satisfy multi-criteria constraints on multiple inputs and outputs. Each time it performs an action, it learns from it, improving its control and adapting itself to the evolution of the process. Moreover, the independence between Controller Agents gives SOC a certain modularity. Each Controller Agent (and its related Context Agents) is a stand-alone MAS that can be plugged on a process input. Regardless on what is controlling the other inputs, it will be able to synchronize its actions to perform a correct control without any knowledge about the rest of the controlling system.

We hope to make SOC generic enough to be easily applied to all kind of systems. It has already been applied to the control of the temperature of a bioprocess (Videau et al., 2011) and is currently being applied in the contexts of ambiant systems (Guivarch et al., 2012), intelligent building energy management and heat engine control.

REFERENCES

Ashby, W. R. (1956). *An Introduction to Cybernetics*. Chapman & Hall, London, UK.

Astrom, K. J. and Hagglund, T. (1995). *PID Controllers: Theory, Design, and Tuning*. Instrument Society of America, Research Triangle Park, NC, second edition.

Colosimo, B. M. and Del Castillo, E., editors (2007). *Bayesian Process Monitoring, Control and Optimization*. Taylor and Francis, Hoboken, NJ.

Feldbaum, A. A. (1960-1961). Dual control theory, I-IV. *Automation Remote Control*, 21-22.

Georgé, J.-P., Gleizes, M.-P., and Camps, V. (2011). Cooperation. In Di Marzo Serugendo, G., editor, *Self-organising Software*, Natural Computing Series, pages 7–32. Springer Berlin Heidelberg.

Guivarch, V., Camps, V., and Pninou, A. (2012). Context awareness and adaptation in ambient systems by an adaptive multi-agent approach. In *International Joint Conference on Ambient Intelligence, Italy*.

Hagan, M. T., Demuth, H. B., and De Jesus, O. (2002). An introduction to the use of neural networks in control systems. *International Journal of Robust and Nonlinear Control*, 12(11):959–985.

Lee, C. C. (1990). Fuzzy logic in control systems: Fuzzy logic controller. *IEEE Transactions on Systems, Man and Cybernetics*, 20(2):404–418.

Nikolaou, M. (2001). Model predictive controllers: A critical synthesis of theory and industrial needs. *Advances in Chemical Engineering*, 26:131–204.

Stengel, R. F. (1991). Intelligent failure-tolerant control. *IEEE Control Systems*, 11(4):14–23.

Videau, S., Bernon, C., Glize, P., and Uribelarrea, J.-L. (2011). Controlling Bioprocesses using Cooperative Self-organizing Agents. In Demazeau, Y., editor, *PAAMS*, volume 88 of *Advances in Intelligent and Soft Computing*, pages 141–150. Springer-Verlag.